Simple Graphics Package

Professor Hugh C. Lauer

CS-1004 — Introduction to Programming for Non-Majors

(Slides include materials from *Python Programming: An Introduction to Computer Science*, 2nd edition, by John Zelle and copyright notes by Prof. George Heineman of Worcester Polytechnic Institute)

Review:– Objects

- ***Object:*** **computational abstraction that includes**
  - Data
  - Methods (a. k. a. Functions)
- ***P. 81:–***
  - **Objects "know" stuff**
  - **Objects "do" stuff**
- **Objects organized into *classes***
  - If ***C*** is a class, then the *constructor* function for that class is also named ***C***
- **I.e., V = `C(arg, arg, …)` creates a new object of class C, assigned to variable V**
- **W = V causes W to refer to very same object as V**
  - Same for assigning object to an element of a list

Why introduce *Objects* so early in the course?

- Because graphics.py is an "object-oriented" package
- Simple enough to introduce now
- A chance to do some cool stuff in your first course in programming!

Simple Graphics Package

- **Described in Chapter 4**
  - Including examples
  - Including exercises
- **Written entirely in Python**
  - Uses existing Python module called Tkinter
    - Based on separate application called Tcl/Tk
- **Conceptually very simple**
  - Programmer-friendly
- **Downloadable from course website**
  - http://web.cs.wpi.edu/~cs1004/a16/Resources/graphics.py
  - Install in folder where you keep your Python programs …
  - … or where IDLE goes by default to open stuff …
  - … (least likely) where Python stores other packages

Examples

- **Game of Life**
- **Simple version of Pong**
- **Homework #3 uses this package**
- **Reading assignment — Chapter 4 of textbook**
  - Read this chapter carefully!!
  - Type out the code on pp 85-86 yourself
  - See if you can get something that looks like Fig 4.3

Components of Graphics system

- **Window**
  - Place in which to draw
  - I.e., a "canvas"

  ```
  win = graphics.GraphWin()
  ```
  - Defaults to 200-by-200 pixels
  - Optional arguments to specify title, width, height

- **`GraphWin` methods to**
  - Get or check mouse clicks
  - Set background color
  - Plot individual pixels
  - Set up coordinate system
  - Close window

- **May have as many windows open as needed!**

Basic Shapes

- Lines
- Circles
- Rectangles
- Ovals
- Polygons
- Points
- Special methods for each

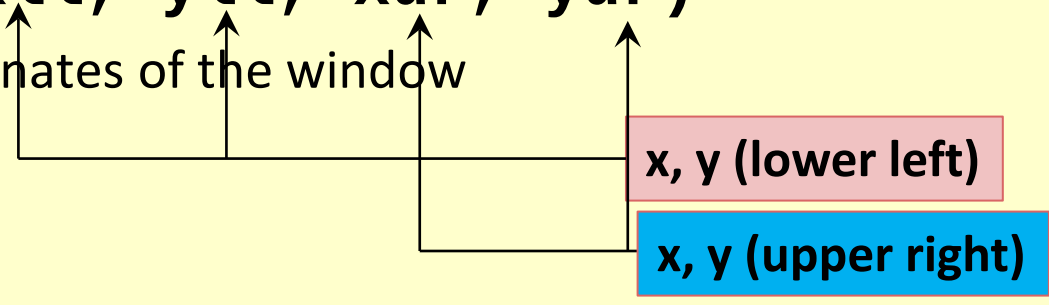
Common methods (for all shapes)

- **SetFill(color)**
  - Color of interior of object
- **SetOutline(color)**
  - Color of the line
- **SetWidth(pixels)**
  - Width of lines
- **draw(window)**
  - Displays the shape in the window
  - Later shapes on top of earlier ones
- **undraw()**
  - Removes from window
- **move(dx, dy)**
  - Moves object in window; redraws if necessary
- **clone()**
  - Creates a duplicate object (not drawn)

Questions?

GraphWin Methods

- **`GraphWin(title, width, height)`**
  - Constructor — creates a window
  - Width and height are measured in pixels
  - Must have a title
- **`setCoords(xll, yll, xur, yur)`**
  - Sets the coordinates of the window



  - Needed so that you can do measurements in "natural" units
  - All subsequent graphics methods measured in these coordinates
    - Positioning of objects
    - Size of objects
- **Needed for Homework #3**

Homework #3

- **Option 1:–**
  - Line and square, plot intersection
- **Option 2:–**
  - Click on three points to draw a triangle
- **Option 3:–**
  - Create bouncing ball

Grading options

- **Option 1 contributes up to 75 points to final grade**
- **Option 2 contributes up to 100 points to final grade**
- **Option 3 contributions up to 150 points to final grade**
- **Choose the option that is best for**
  - Your comfort level with programming in Python
  - The level of work you can invest in this Homework

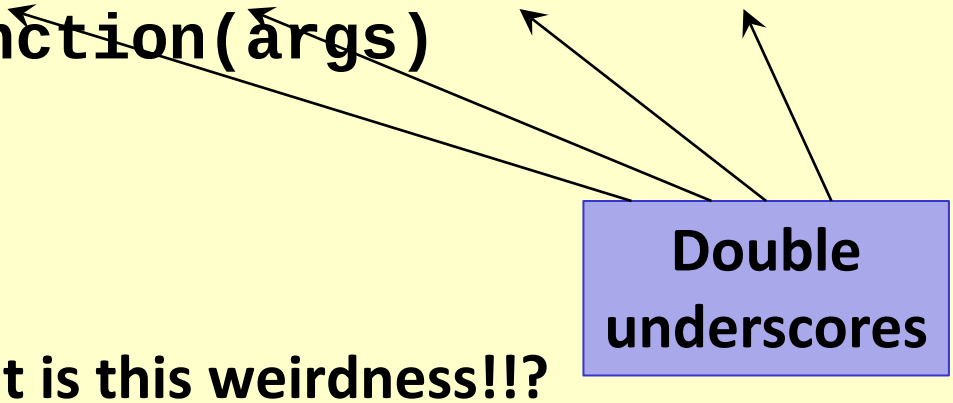
Organizing your program

- **Write you program as a single .py module**
- **Define functions**
- **(Optionally), define any global variables that are needed**
- **(If selecting Option 3):–**
  - Read ahead to learn about **`if-else`** statements (p. 201)
- **Add special code at the bottom to execute the main function of the module**
  - But only if you are running this module, not importing it into something else

Special code to start your module

- **At the bottom of your .py file, include the following:–**

```
if __name__ == '__main__':
   function(args)
```



- **What is this weirdness!!?**

- **Answer:–**
  - Every module has a name
  - Name is stored in magic variable "**`__name__`**"
  - When you *Run* the module **(as opposed to importing it)**, …
  - … *Python* changes its name to "**`__main__`**"
- **Result:–**
  - In a multi-module program, put this at the bottom of the module to tell *Python* where to start